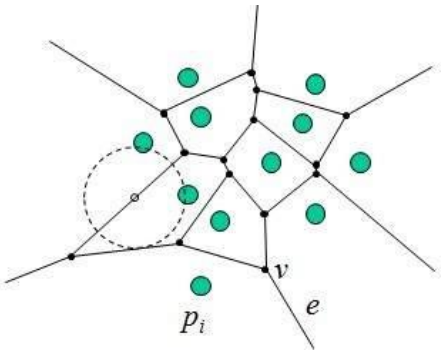
Voronoi 图扫描线算法的三维演示

1. 最近 Voronoi 图定义及性质

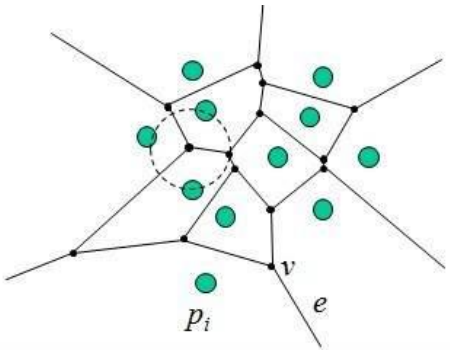
Voronoi 图的定义：

在平面上有 N 个独立的站点$p_i, i = 1,2 \ldots\ldots, N$，而 Voronoi 图就是把平面分成 N 个子区域，每个站点都拥有自己的子区域，在这个区域中的任何点 q 到当前站点的距离比到其他站点的距离最短。

Voronoi 图的性质：



图一



图二

如图一所示，站点p_i与p_j对应的 Voronoi 边上的点在p_i与p_j的垂直平分线上，以这个点为圆心的圆能够经过p_i与p_j并且圆内无其它站点。

如图二所示，如果一个点是 Voronoi 定点，则它至少经过三个站点，并且圆内无其它站点。

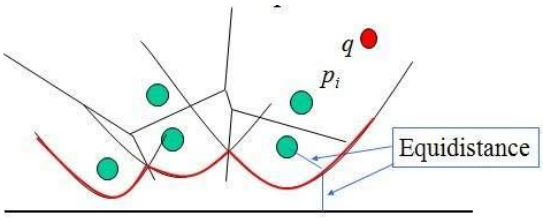
2. Voronoi 图扫描线算法

扫描线算法概述：

1. 通过水平线从上往下扫描站点；
2. 增量构造，跟踪每个站点对应的结构的变化。

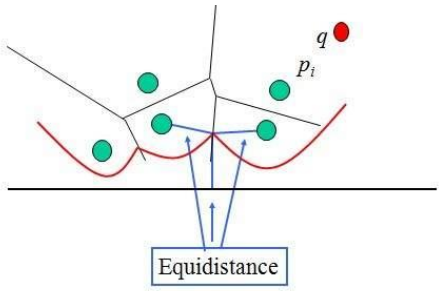
扫描线算法待处理事件：

1. 如图三所示，图中的红色弧的序列为海岸线，是我们要跟踪处理的数据结构（组织成二叉树）。



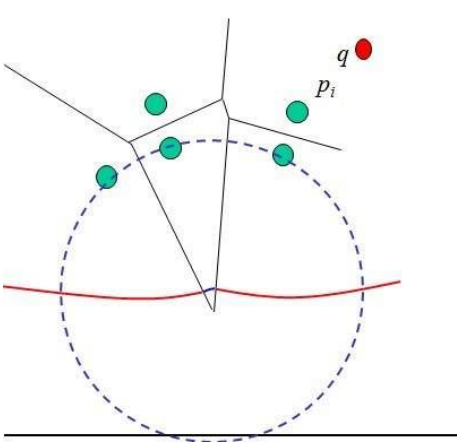
图三

2. 图四中到两个站点及扫描线相等的点为分裂点，为海岸线结构中的重要成分，实际上为二叉树中的内点，而每条弧则为叶子节点。

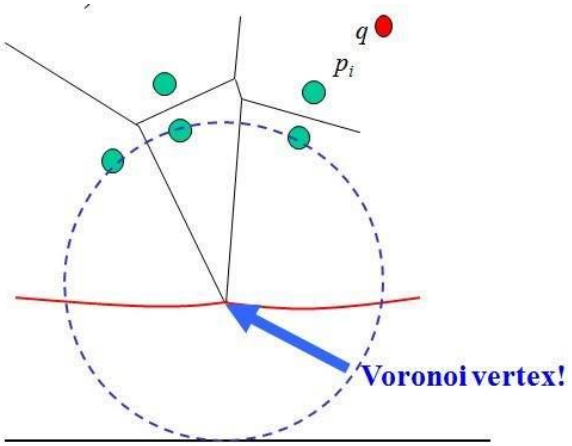


图四

3. 图五和图六为两个连续的瞬间，图五中间的那条弧即将消失，取而代之的是 Voronoi 顶点。它的两条边为分裂点生长而成。



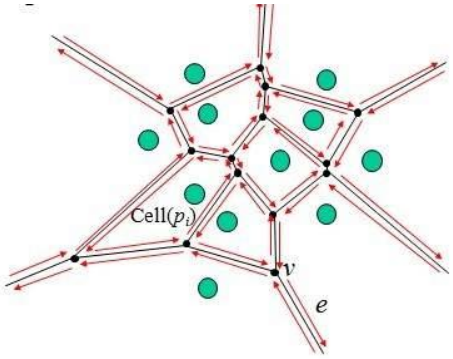
图五



图六

我们所用的数据结构：

1. 用 DCEL 记录 Voronoi 图：



图七

Vertex：

点的辅助信息

bool inner; 表示该点是不是一个内部点（非边界点）

vector<int> inTris; 记录该点所在的三角形号

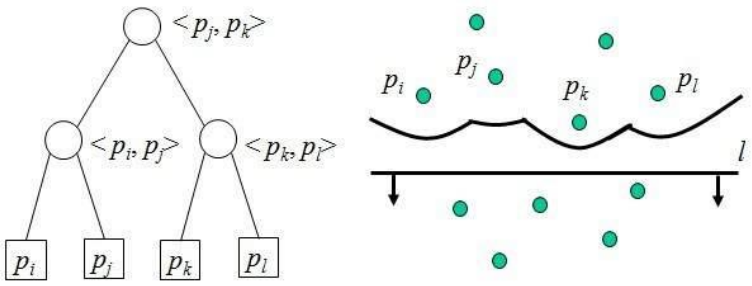
vector<int> inTrisOrd; 记录该点在相应三角形中的编号（只取 0，1，2）

int startHe;　该点起始半边编号
int endHe;　　该点终止半边编号（仅对边界点有效）

HalfFace3：
面辅助信息
int he[3]; 记录一个面中三条半边号码

HalfEdge：
基础半边结构
int fv;　　起始点
int tv;　　终止点
int fn;　　面号
int prev;　前一条半边
int next;　后一条半边
int opp;　对面相反的半边

2. 海岸线结构，也即是二叉树。树中的内节点为两条弧相交的分裂点；叶子节点为它们所代表的弧，与生成弧的站点相对应，注意同一个站点有可能出现两次，被其它弧所割裂。

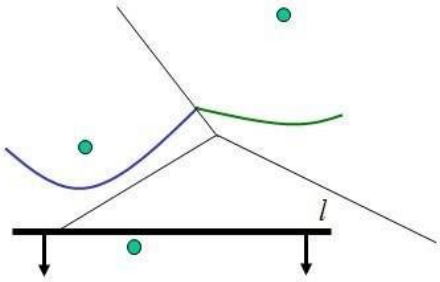


图八

定义基类 node
内部基类 : interior_node
left_index ;　左边节点的叶子编号
right index;　右边节点的叶子编号
叶子基类 :leaf_node
cycle_event*: 指向一个圆事件
index: 叶子所在的站点

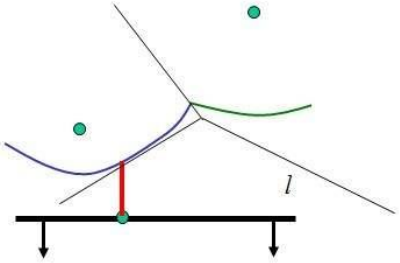
3. 事件队列，包含所有站点事件和圆事件，它们按照 y 坐标排序。

站点事件：
站点事件前



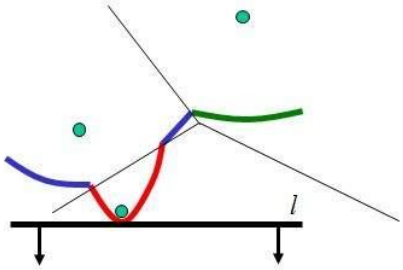
图九

当扫描线刚刚经过一个新的站点，则新的弧被创建



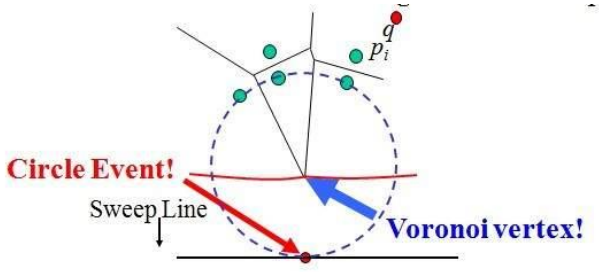
图十

在新的站点上原来的弧被一分为二



图十一

圆事件



图十二

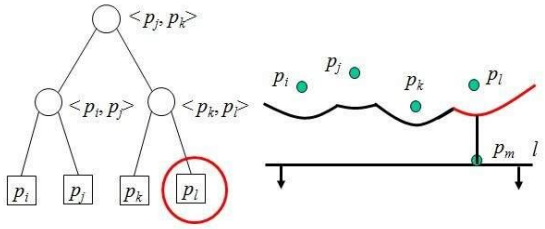
如果一个圆能经过三个以上的站点，并且圆内无其它的站点，则相应的弧消失，转化成圆的圆心。扫描线与圆的切点为圆事件点。

扫描线算法：

```
1. Initialize
    • Event queue Q ← all site events
    • Binary search tree T ← ∅
    • Doubly linked list D ← ∅
2. While Q not ∅,
    • Remove event (e) from Q with
      largest y-coordinate
      HandleEvent(e, T, D)
```

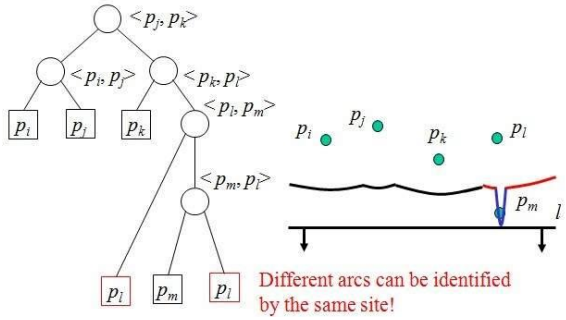
当 e 为站点事件时：

1. 定位此站点为在哪条弧下面（根据 x 坐标）



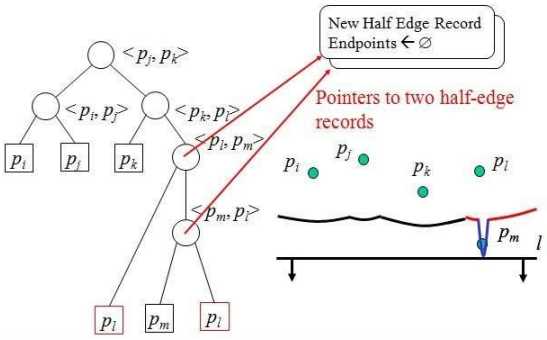
图十三

2. 将定位到得弧一分为二



图十四

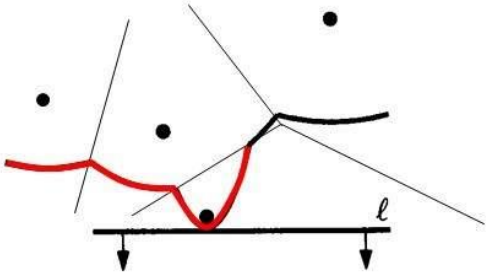
3. 添加新的半边到 DCEL 结构中，endpoint 为空



图十五

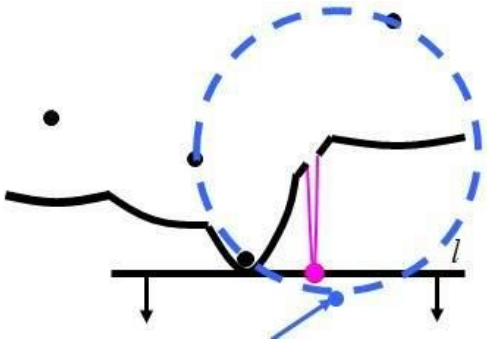
4. 预测目标圆事件。扫描相邻的三段弧，看是否能产生圆事件。（当新的弧在中间的时候，

不可能产生圆事件）



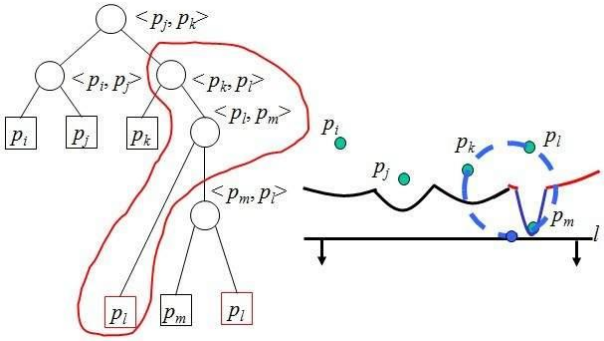
图十六

注意：并不是所有分裂点相交的情况都能生成圆事件点，如图十七所示。所以我们要注意判断这种情况。



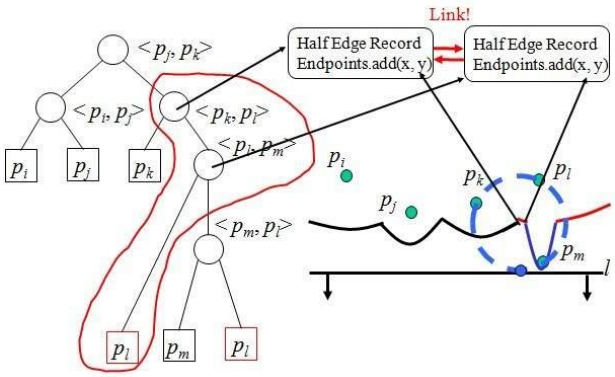
图十七

当 e 为圆事件的时候：



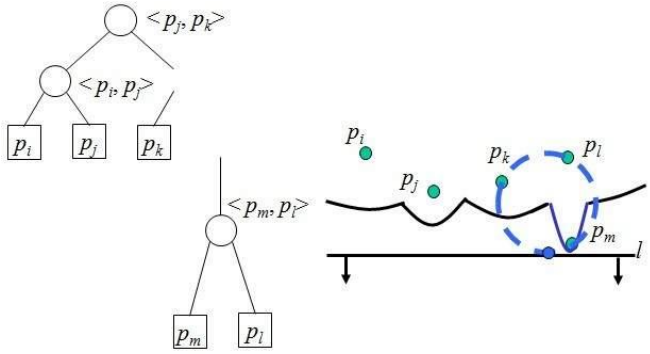
图十八

1. 把将要消失的弧转化成 Voronoi 顶点加入半边结构中，并且把原来内节点的半边的终点赋值成该顶点。

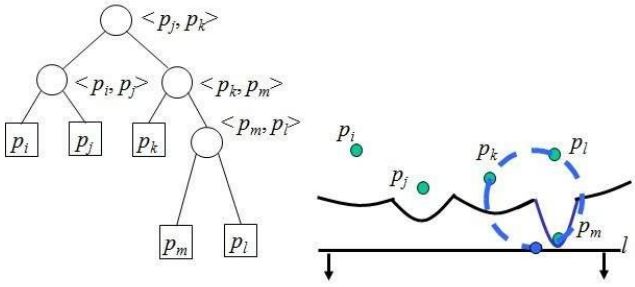


图二十

2. 删除消失的弧

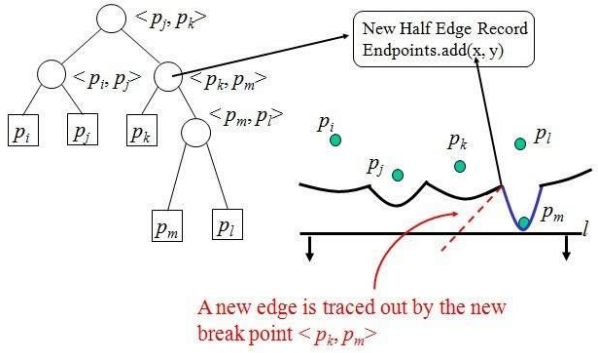


图二十一



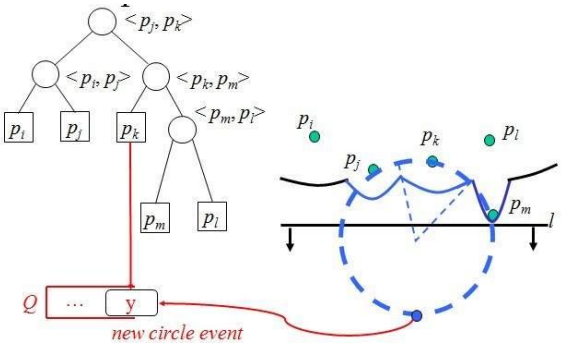
图二十二

3. 创建新的以上面顶点为终点的半边



图二十三

4. 看是否还能生成新的圆事件



图二十四

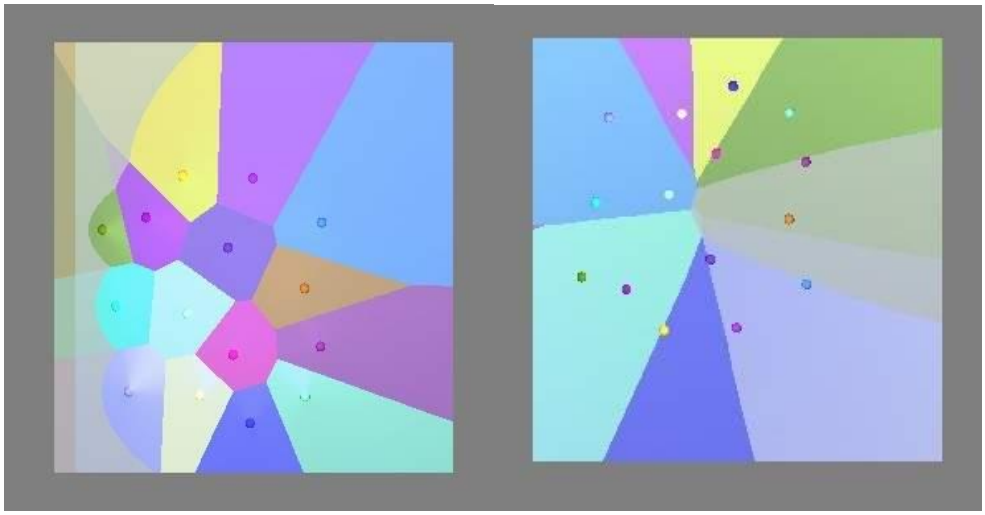
3. 界面说明及实验结果

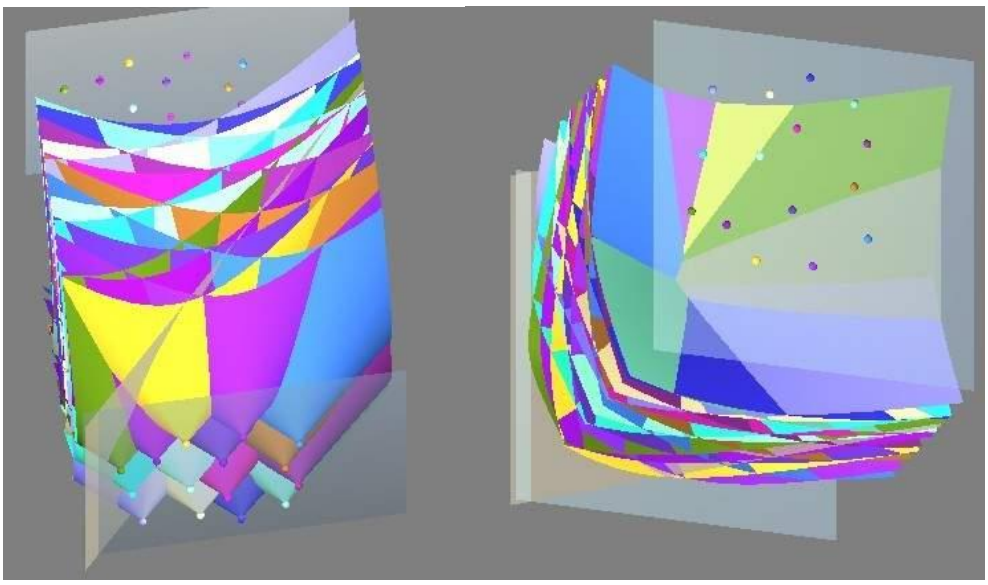
如图所示，左边子窗口为根据所给的站点生成的 Voronoi 图，这个 Voronoi 图是最近 Voronoi 图。左边子窗口为把最近 Voronoi 图投影到三位坐标上的情况，以站点为圆锥顶点，生成半径大小相等的圆锥，生成的交界曲线在站点平面上的投影即为最近 Voronoi 图的边。在另一面上的投影即为最远 Voronoi 图。以夹角为 45 度的平面为交平面，它与站点平面的交线为扫描线，与圆锥的交线投影到站点平面则为海岸线也即是左边子窗口的红线。

界面使用说明：

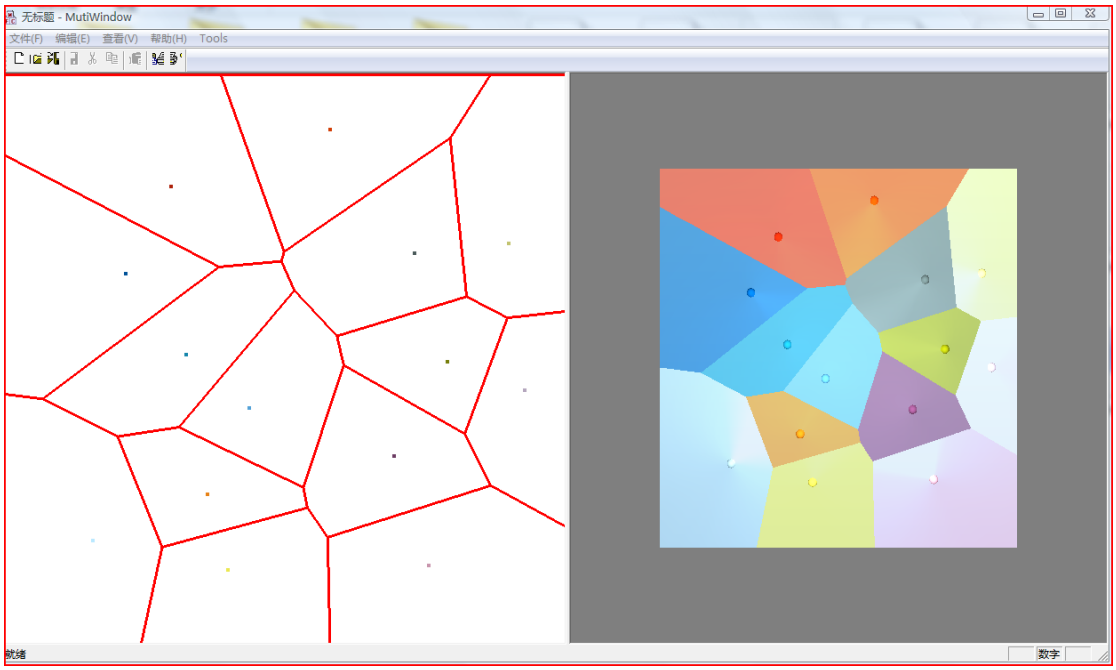
左边子窗口：点击鼠标，在左右窗口同时生成站点（右边同时生成对应圆锥）。

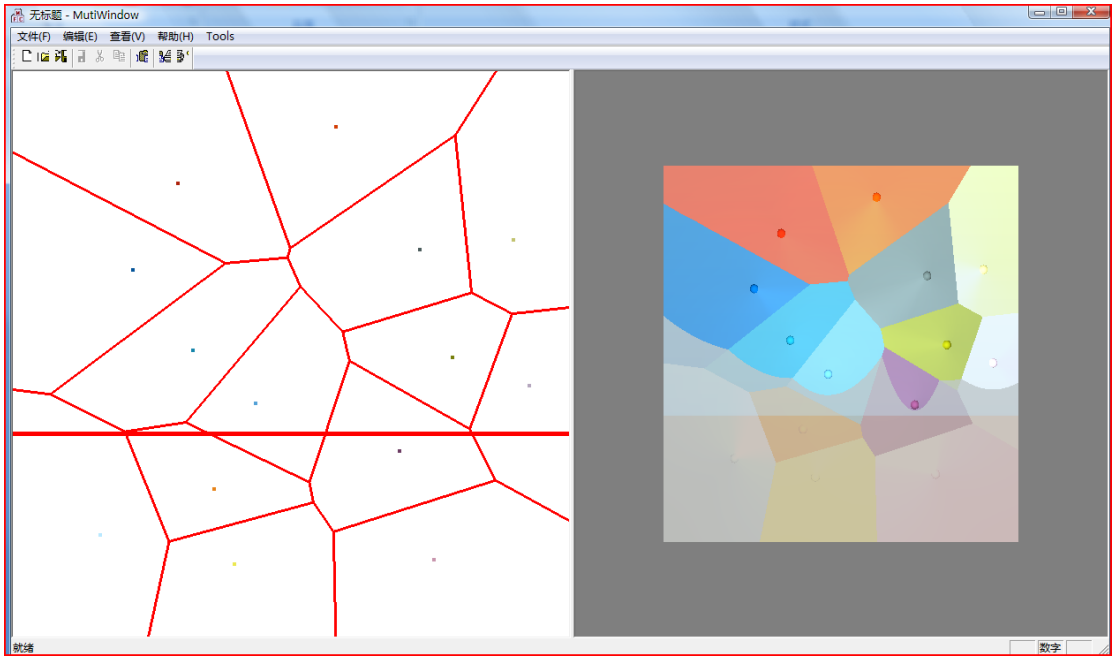
右边子窗口: 1. 拖动鼠标则可以查看 Voronoi 图的三维细节；2.按住 shift 键，拖动鼠标则可以放大缩小三维 Voronoi 图；3. 按住 ctrl 键，拖动鼠标则可移动三维 Voronoi 图；4. 按->键能够移动相交平面以及对应的扫描线，按住 ctrl 和->能加快移动速度；5. 点击鼠标右键，选择 render，然后在选择 show cut plane,通过移动左右方向键，就可以动态的演示扫描线算法，然后点 Voronoi 则可查看最近 Voronoi 图。点 Farthest Voronoi 则可查看最远 Voronoi 图。6. Reset View 重置视角， Clear Points 清空现有的点。

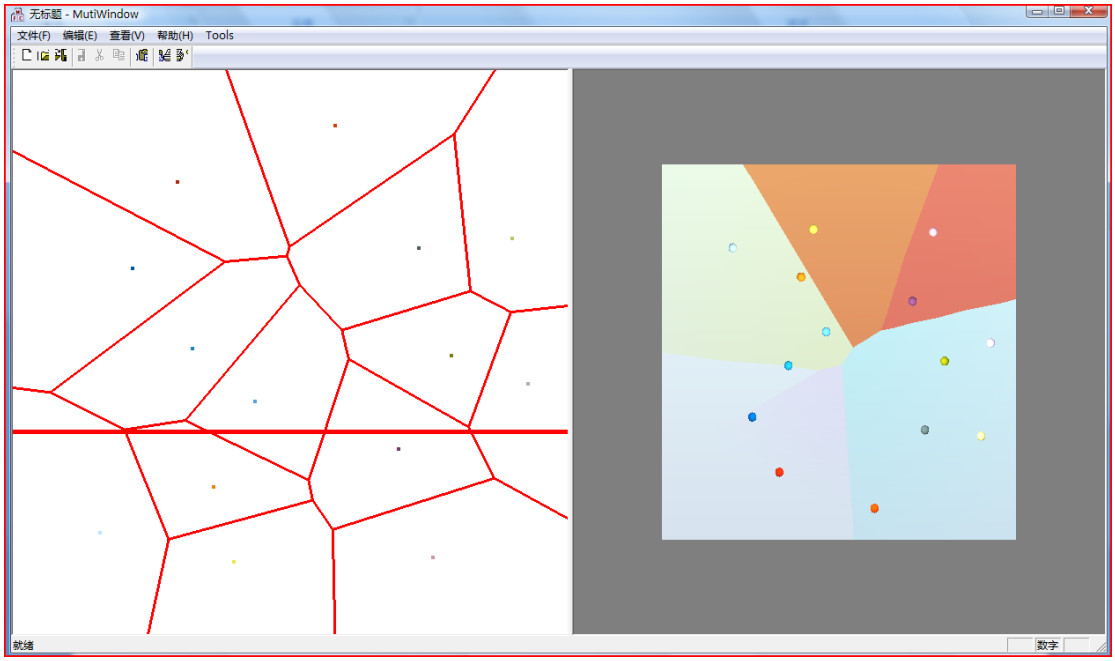


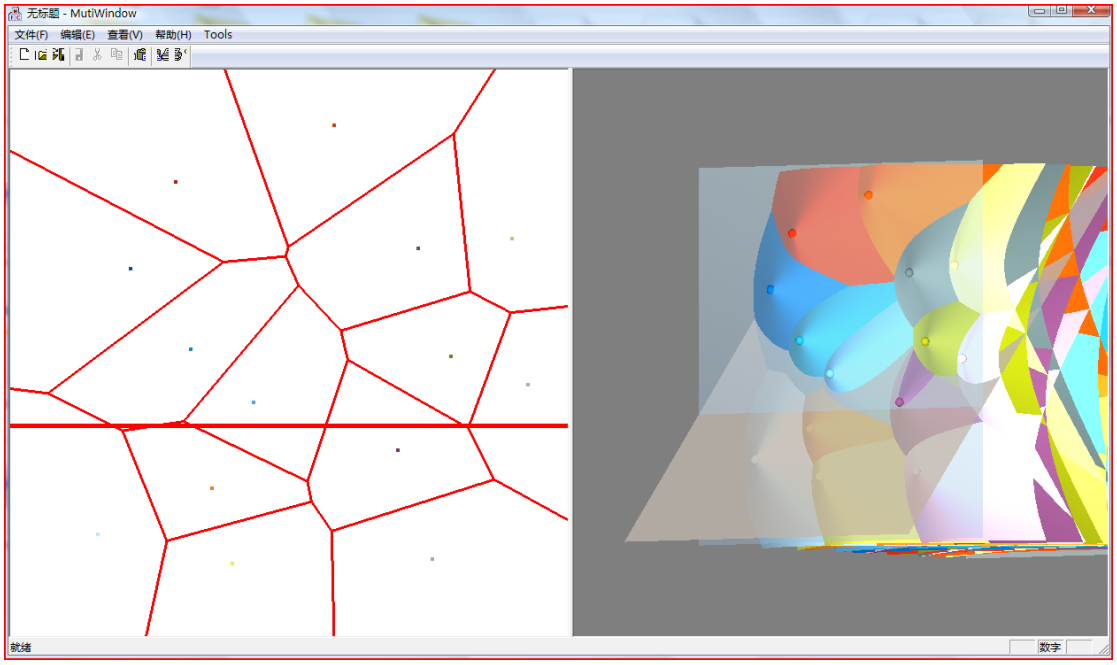


以下是程序运行结果截图：




无标题 - MutiWindow
文件(F) 编辑(E) 查看(V) 帮助(H) Tools
就绪
数字


无标题 - MutiWindow
文件(F) 编辑(E) 查看(V) 帮助(H) Tools
就绪
数字



算法局限和待解决问题

由于时间比较仓促，有时会出现 bug，在点多的时候，使用平衡查找二叉树会提高算法的效率，但是由于其很复杂，我们这里就没有使用平衡查找二叉树，在点少的时候，效率反而不一定高。

参考文献：

[1] S. J. Fortune. **A sweepline algorithm for Voronoi diagrams, Algorithmica**, 2:153-174, 1987.

[2] Junhui.Deng. **Computational Geometry Course 2006, Lecture Notes**. Chapter 4, Chapter5